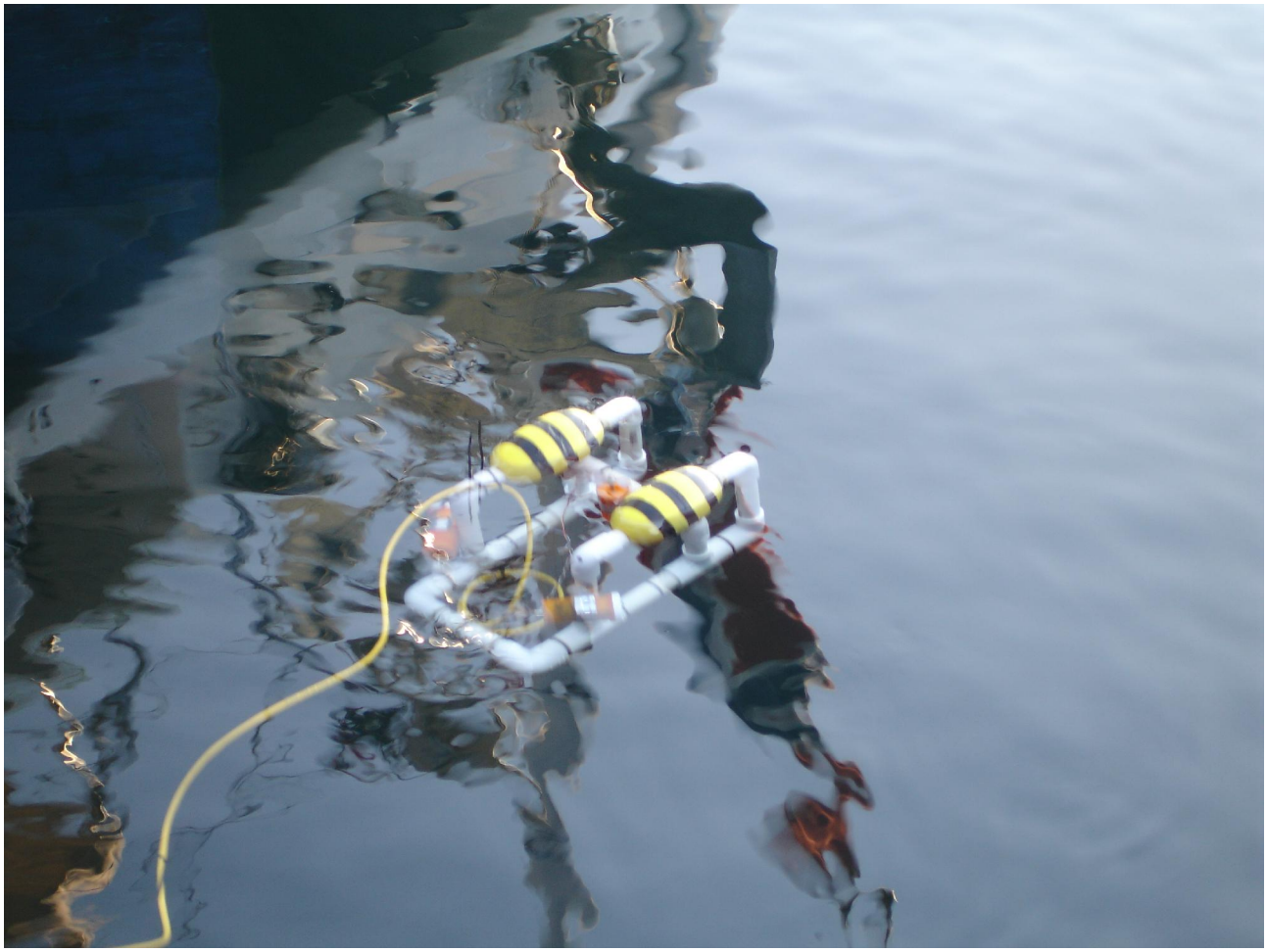
Open Hardware Journal

November 1, 2011 – **Open-Access Journal**, free to read, copy, and redistribute.



Sea Perch – a Remotely-Operated Underwater Vehicle, easily built and operated by middle-school students from common and inexpensive parts, developed by the MIT Sea Grant College Program.

Contents

Open Hardware Journal

Published by the *Open Hardware* organization. Please see our web site at OpenHardware.org

Editor: Bruce Perens <bruce@perens.com>

***Open Hardware* means sharing the design of physical or electronic objects with the public, similarly to Open Source software. The right to use, modify, redistribute, and manufacture, commercially or as a non-profit, is granted to everyone without any royalty or fee. Thus, Open Hardware designers hope to enrich society by developing a library of designs for useful objects that everyone can make, use, and improve.**

Editorial: Some Great Open Hardware Projects

By Bruce Perens <bruce@perens.com>

There's a lot of excellent Open Hardware that you might not have heard of. On the cover, and discussed in a paper in this issue, is *Sea Perch,* a remotely-operated underwater vehicle for science and technical education. Sea Perch is easily built by middle-school students, from inexpensive parts like PVC pipe. You can get a copy of the design from http://SeaPerch.mit.edu/ or a kit of parts from http://SeaPerch.org/

While teaching at *University of Agder,* I guided a class of Ph.D. Students and teachers through building the Sea Perch. I'd purchased an inexpensive underwater TV camera, intended for fishermen, and taped it on a Sea Perch. We sailed the sub around a dock in the Grimstad, Norway harbor, viewing the underwater scene through a color TV on the dock. Norway's above-water beauty was matched by our little ROV's viewpoint, displaying schools of fish and aquatic plants.

A Gray-Hoverman antenna built from Jeff H's design.

I was surprised to find that PVC pipe was 10 to 20 times as expensive in the local Norwegian hardware stores as in my Home Depot store in California. I ended up importing a suitcase full of pre-cut segments.

Once you've acquired the skill of constructing with PVC pipe from *Sea Perch,* a suitable next project would be the Gray-Hoverman TV antenna, which might help you receive some distant HDTV stations.

The best method of building one from PVC pipe that I've seen was designed by “Jeff H.”, an engineer for WSYR Television. You can get his plans in these three files:

http://www.centralmediaserver.com/WIXT/Engineering/grayhovermanantennapartslist.pdf

http://www.centralmediaserver.com/WIXT/Engineering/antennapipecuttingdetail.pdf

http://www.centralmediaserver.com/WIXT/Engineering/gray-hovermanantennadrawing.pdf

Surprisingly, when I went to purchase the PVC “cross” fittings called for in Jeff's design, which connect 4 pipes together, I found that the use of them in plumbing wasn't authorized in California. Thus, my local Home Depot didn't stock the cross, and refused to ship the part to California when ordered online. The “tee” fitting, connecting 3 pipes, was easily available. I ordered the cross fittings from another state, but a replacement of Jeff's design for California users might be desirable.

Between *Sea Perch* and *Gray-Hoverman,* I've had to deal with a surprising lot of differences in the geographical availability of PVC. But this is the sort of issue that Open Hardware developers are likely to face. Parts are too often made of “pure unobtainium”, and half of the achievement can be just getting the right pieces together to build your project. The best Open Hardware designers work hard to create a bill-of-materials that people can buy easily, and many end up producing parts kits to spare the builder the effort of locating components.

http://wiki.openhardware.org/Category:Vendor is a good place to share information about where you've found parts.

If *Genetic Engineering at Home* piques your interest, look at http://OpenPCR.org/ where you can find the design for a thermal sequencer for the polymerase chain-reaction, or a parts kit. PCR is a *DNA amplifier,* a key tool for genetic analysis or even perhaps unicellular genetic engineering by the motivated individual. The other tools necessary, for example gel electrophoresis columns, are also within range of the dedicated hobbyist. We hope to have an article from the OpenPCR folks in the near future.

HPSDR is a project to buld a high-performance software-defined radio transceiver, for use by ham radio operators and other radio experimenters. Theirs is a multi-card design with a backplane, and many of the card designs have been manufactured and sold with the financial and technical support of TAPR, the leading organization for the development of digital ham radio. HPSDR formerly used a non-commercial license for some designs, but they are transitioning to exclusive use of an Open Hardware license for TAPR-sponsored boards. You can find the designs and parts kits at http://OpenHPSDR.org/

Want your Open Hardware project to be better known? We're developing an *Open Hardware Catalog* at OpenHardware.org . Please add your project, start with the instructions at http://wiki.openhardware.org/Project:How_to_Create_Content_for_the_Open_Hardware_Catalog

Open Hardware Journal is brand new. We'll need your help to get through the next few issues: not financially, but in the form of articles, and we need them on a tight schedule. Please see the *Call for Papers* in this issue.

Producing Lenses With 3D Printers

Invited Paper

By Christopher Olah <chris@colah.ca>, http://hacklab.to/

Abstract

A technique for producing optical quality lenses with 3D printers is explored. 3D printed positives of the lens are interpolated with plastic wrap and then used to produce molds. Polyester casting resin is then used to produce lenses. Low-quality lenses are produced, and the causes of failure are discussed.

Keywords: 3D printing, optics, lenses, casting

Introduction

The desire to produce optical equipment with 3D printers arises for a number of different reasons. The author, having produced an experimental surface-oriented CAD program, was personally drawn to it as an application that needed precise non-planar surfaces. Saner motivations include getting components of specific dimensions for projects, creating non-traditional components for experimentation, or simply the joy of having made the lenses yourself. And while one can hand make lenses and mirrors, as is popular in amateur telescope making, sanding glass into the desired shape is very time consuming.

It may seem at first that it would be easier to 3D print mirrors, smoothing them with some sort of post-processing (eg. sanding or coating) and then turning them into mirrors. Both of these, the author discovered, are easier said than done.

While a 3D printed object can be substantially smoothed by sanding, small cracks tend to remain problematic, at least for FDM plastic objects; furthermore, at least with ABS plastic objects, sanding is somewhat tedious because the dust is an irritant to the lungs and precautions must be taken. On the other hand, coatings have the unfortunate tendency of pooling in regions, causing the surface to cease to be the desired one and nullifying the reasons for 3D printing in the first place.

But the real problem lies in actually turning them into mirrors, since the obvious approaches like spray-painting aren't able to reach adequate quality while silvering and aluminizing require chemical skill, equipment, and involve hazardous chemicals.

On the other hand, products for casting transparent objects are readily available thanks to the artistic crafts community. This makes producing lenses an appealing approach.

Procedure

The author considered and even experimented with a number of approaches for casting the lenses, including a sort of inverse-lost-wax molding with water soluble plastic and directly using 3D printed molds, before settling on using 3D printed positives to make molds and then using those. This decision was made for a number of reasons:

1. There was more opportunity for interpolation to correct for the poor resolution of the 3D printer.
2. A soft mold medium made objects easier to remove

3. A single print can be used to produce numerous lenses

Producing Lens Positive

The first thing that needs to be done is producing a positive model of the desired lens. The following Python code will produce a plano-convex lens in Surfcad:[1]

```
from surfcad import *

F = STLFile("lens.stl")
parabola_top = circular_surface(lambda r,t: 9-r**2/75, 15)

side = cylinderical_surface(lambda t,h: 15, 6)

F.add(parabola_top.surface() )
F.add(side.surface() )
F.add(side.bottom_loop().close())

F.end()
```

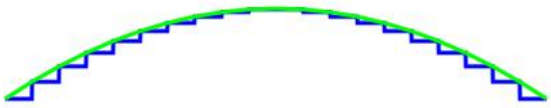
Or this Haskell code in the author's most recent CAD project, ImplicitCAD:[2]

```
import Implicit

out = intersect
        (zsurface \(x,y) -> 9-(x^2+y^2)/75 )
        (cylinder 15 10)

writeSTL (-16,-16, -1) (16,16,11) 1 "lens.stl" out
```

Once the model is printed, a bit of post processing in the form of sanding makes sense.

Interpolation

Even the lower end of optical precision is a high-standard, and 3D printers can't directly reach it. Some form of interpolation is essential.



As it turns out, pulling plastic wrap over the surface goes a long ways to improve the quality of the surface because it performs linear interpolation between a hundred or so points over a few centimeters, This is a fairly good approximation of a smooth, slow changing curve, whereas the layer by layer construction of most 3D printers isn't anywhere close.

A thin layer of oil can prevent wrinkles from forming if you need to apply multiple layers of plastic wrap. Furthermore, in some types of wrap that don't really cooperate with interpolation (the one the author experienced this with is unbranded), a film of oil will. The oil can also act as a mold release.

---

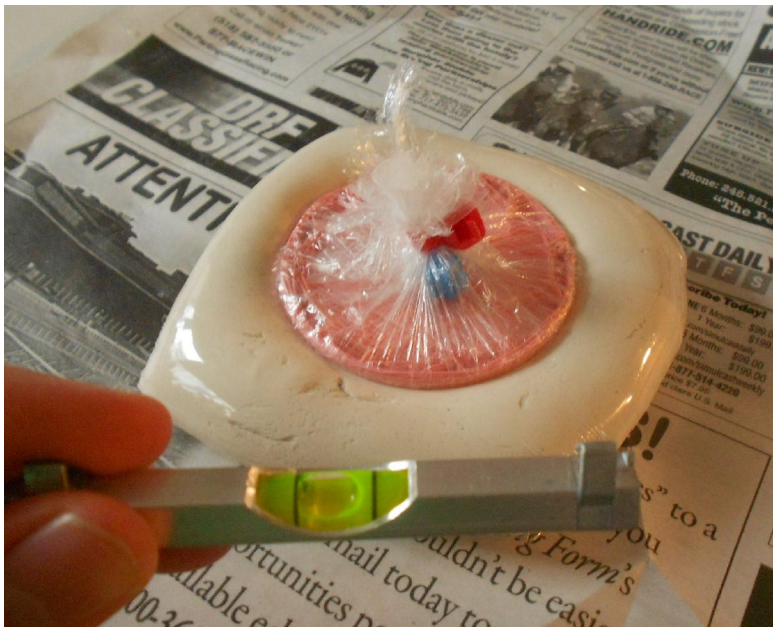
1 http://github.com/colah/surfcad

2 http://github.com/colah/ImplicitCAD

Making the Mold

The author experimented with using plastic wrap coated clay and silicone putty for the mold. While the silicone putty is much easier to work with, the results of the plastic wrap are at least as good, if not better because of improved interpolation.

Either way, the mold medium is prepared and the interpolated positive impressed into it. It is important to make sure that the mold is level, but also not to fidget trying to fix it since that will cause wrinkles.



If the material needs time to set, don't take it out early: damaging your mold isn't worth a few extra minutes.

A mold-release compound should be applied during this process, so that the positive can be removed without damaging the mold.

Casting the Resin

The author tried several epoxies and resins, having the most success with *Castin'Craft CLEAR Polyester Casting Resin.* In addition to being much clearer than the other ones we tried, it was less viscous than the others, which helps prevent bubbles. It seems highly probable that there are other, better, choices, that were not tested.

Most epoxies and resins need to be stirred in the preparation process; it is very important at this stage to avoid making bubbles. In some epoxies, bubbles will burst on contact with CO^2, so holding one's breath and then breathing on it will pop them.

Once prepared, slowly poor your resin on the edge not shifting -- this avoids bubbles creases on the surface.

It is important to note that epoxies and resins can give off noxious gasses. Read the included safety information/MSDS.

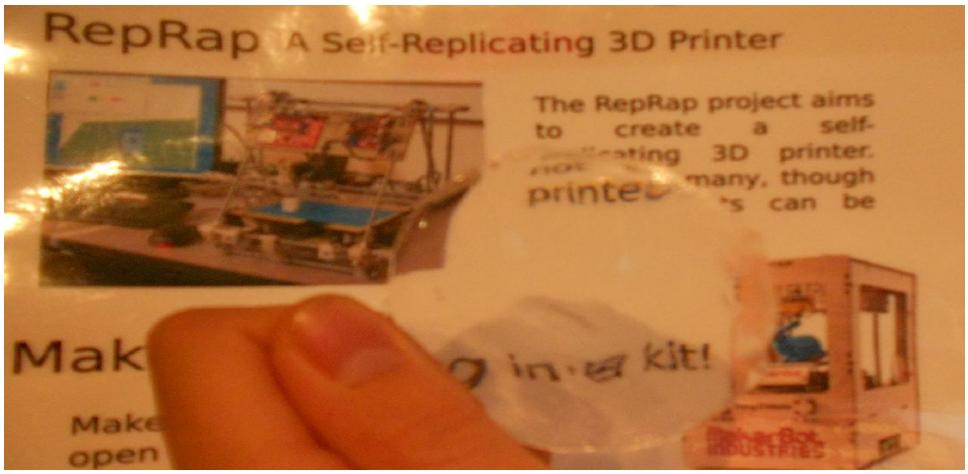
Post-Casting

Gently remove your object from the mold. If you cast into plastic wrap, not removing the wrap is an option to consider

Depending on your resin/epoxy choice, your lens may be soft and sticky for days afterwards. This bodes caution. Be careful where you set it, and make sure others understand that they should hold you lens by the edges *before* you pass it to them.

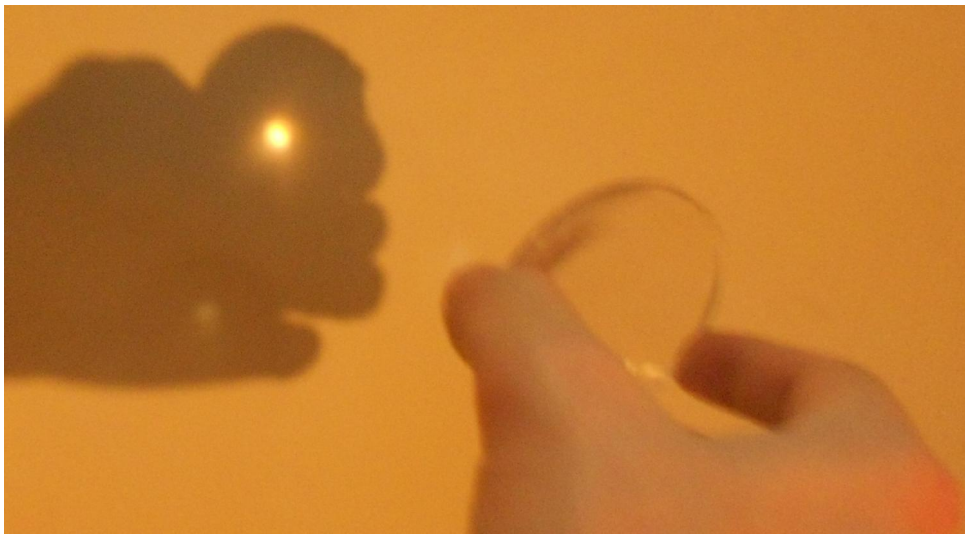
Results

While the author's results are not yet comparable to the results of commercial lenses, they are definitely lenses.



The lenses can focus parallel light to a certain point, though flaws prevent one from reaching the pin-prick focus a commercially produced lens can achieve.

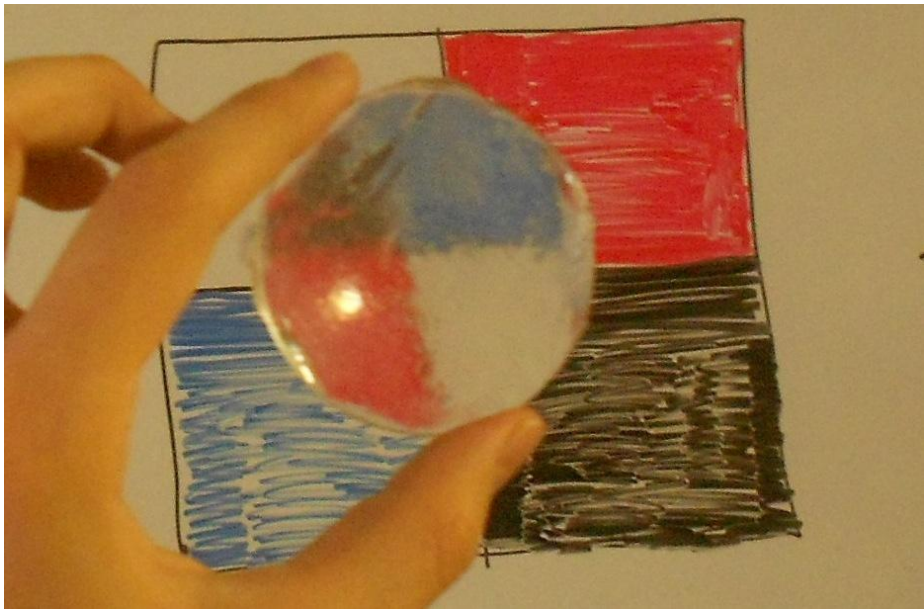
More rigorous tests reveal flaws, however.



While the image is flipped as expected, there is blurring and imperfections. The lens used in the above image has rings from poor interpolation and blurring from tiny bumps on the curved surface. While these flaws vary in form and cause, the author has been unable to avoid them completely.

A number of modes of failure were mentioned as warning in the Procedure section, but that was not comprehensive and a list seems useful.

- Bubbles are caught in your epoxy/resin. Avoid this by stirring carefully and pouring slowly.
- Casting material dripping out of your mold as it sets causes ripples on the surface. This is more of an issue when the liquid is very viscous.
- Air flow or vibration can also cause ripples.
- Premature removal of the lens from its mold can result in damage to the surface.
- Droplets of mold release can create flaws in surfaces.



- Setting lenses with `tacky' surfaces on most other surfaces will damage them.
- Finger prints can easily form if a soft lens is not handled from the edge.

Additionally, if one is using a plastic wrap coated mold, there is the additional concern of the plastic wrap popping up during setting. Depending on how long the lens has had to set, it may bend and wrinkle or pop out of the mold -- both are problematic.

Conclusion

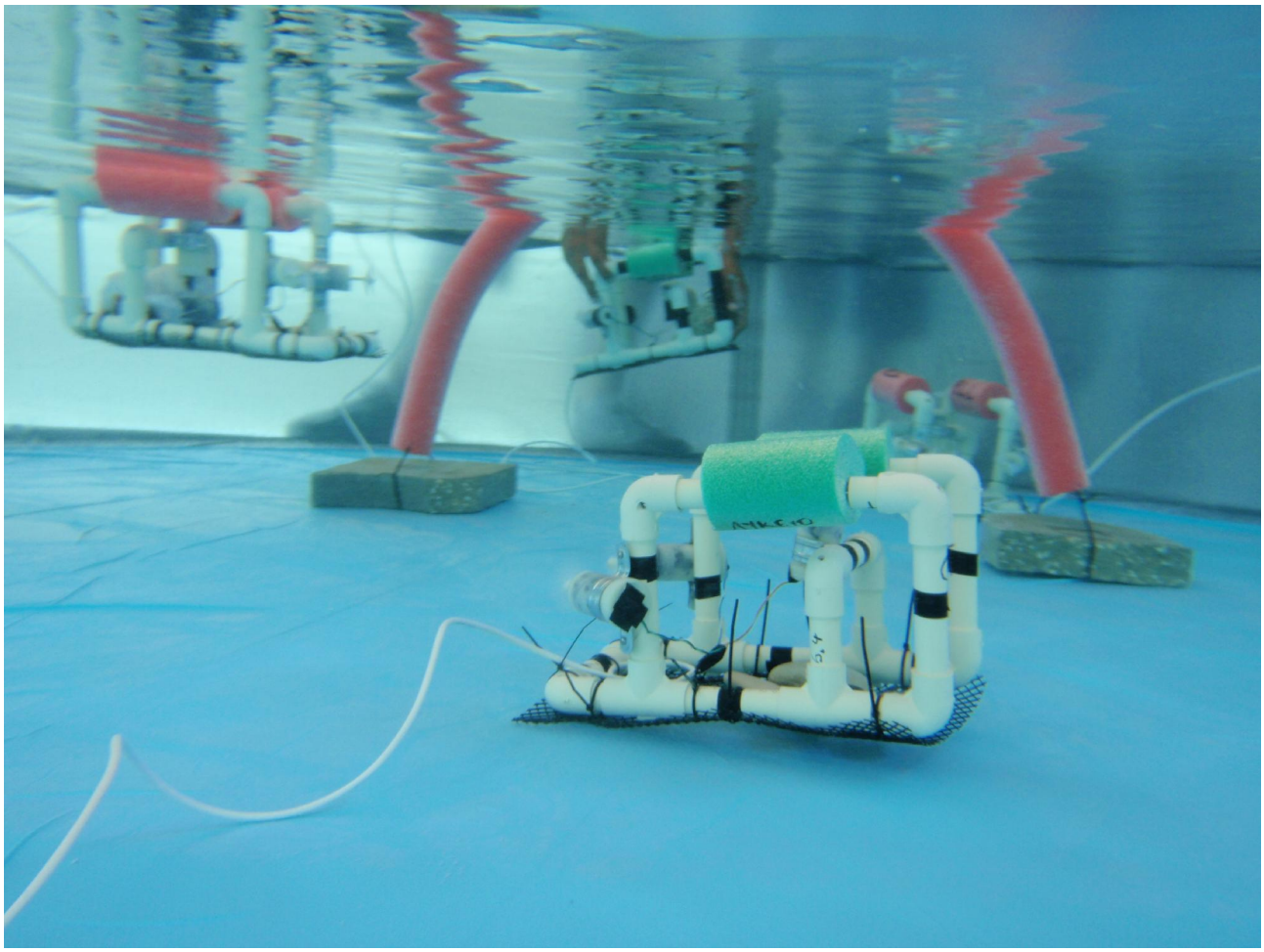
While the lenses that have been produced with the outlined technique are incomparable to commercial optics, and fall short of the quality needed for most visual applications, they are of sufficient quality to be used for some low-precision light distribution applications like collimating light in a flashlight. Furthermore, the wide variety in modes of failure is reason to believe that much higher quality can be achieved, since each one, evidently, can be defeated individually. It is simply a matter of perfecting the technique.

The author intends to pursue this until he can 3D print a telescope...

STEM Education through Open Hardware at MIT Sea Grant

Invited Paper

By Rachel VanCott, Brandy M. Wilbur, Chryssostomos Chryssostomidis, Michael Soroka, and Kathryn Shroyer



This early-stage Sea Perch ROV utilized foam cylinders for ballasting.

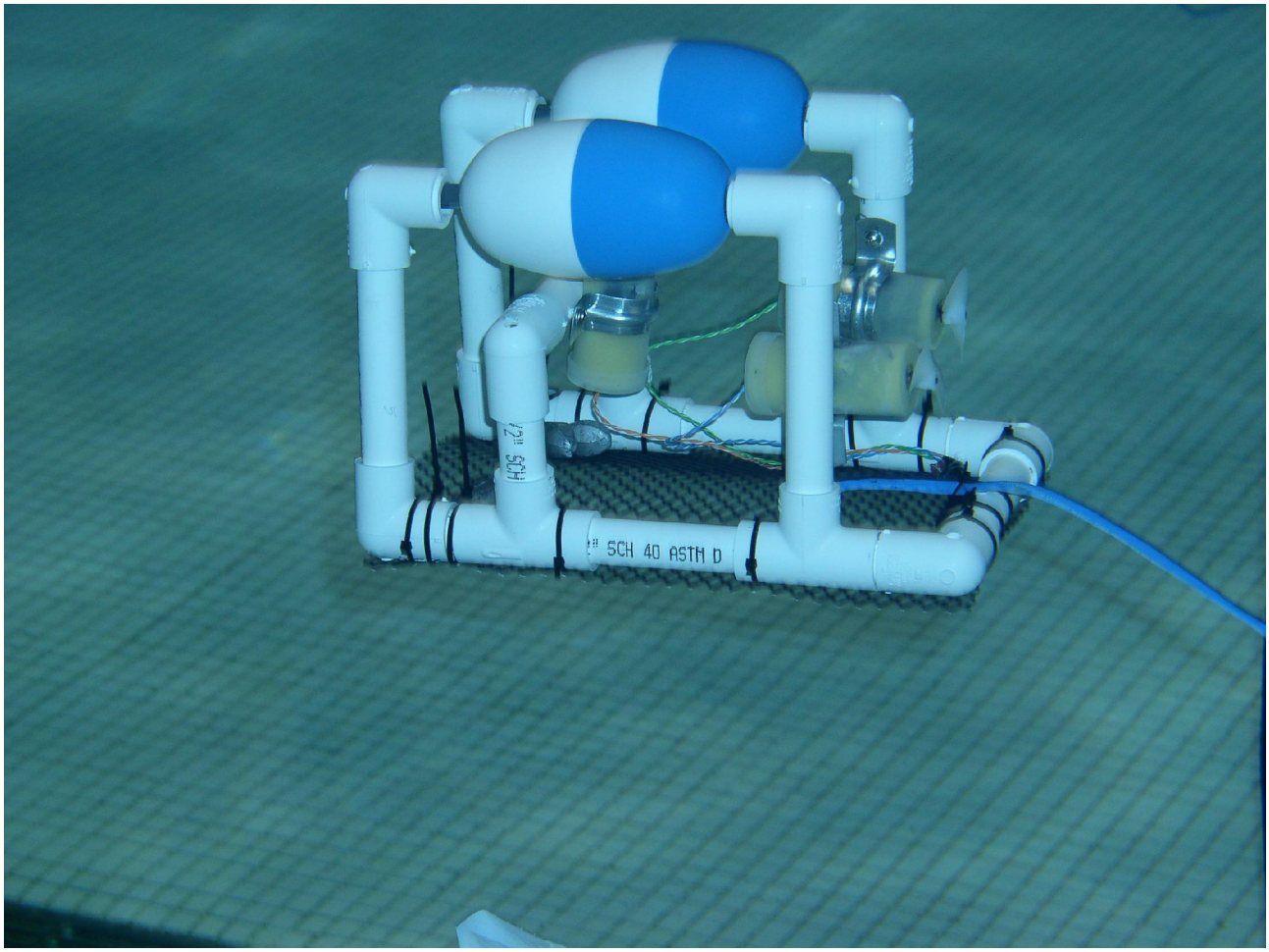
History

The MIT Sea Grant College Program recognizes a need to encourage students of all ages to develop skills in marine science and ocean engineering, and consider advanced study and careers in those fields. To that end, with support from the Office of Naval Research, MIT Sea Grant offers project-based classroom activities and programs that build on the construction, design, and use of a simple and inexpensive remotely operated vehicle (ROV), known as Sea Perch.

The concept for the Sea Perch ROV was first proposed in the 1997 book, *Build Your Own Underwater Robot and Other Wet Projects,* by Harry Bohm and Vickie Jensen. In 2003, with the

authors' permission, MIT Sea Grant transformed the basic outline of a PVC-pipe-based vehicle into a full-fledged build process, and began offering teacher trainings. Since then, MIT Sea Grant has started developing novel Science, Technology, Engineering and Mathematics (STEM) teaching technologies that inspire students to explore marine sciences, engineering, history, and physics, and structured programs that offer an introduction to engineering skills and thinking.

Part of the program's success can be attributed to the "open platform" ethos of the Sea Perch ROV programs. Teachers and students that participate in MIT Sea Grant education projects are provided with full instructions on the basic construction and use of the ROV, but all users are strongly encouraged to modify and extend the function of their projects beyond the introductory level.



Later Sea Perch ROV builds utilized plastic floats, which can travel to greater depths without compressing.

MIT Sea Grant's Sea Perch programs have been effective at igniting school children's enthusiasm for science, technology, and engineering. The program has been a successful tool. We estimate that

our program has reached over 36,000 students, over 300 teachers, and several schools from other countries. Beyond our immediate reach, we also introduced the project to a group that became a partner organization to us--SeaPerch.org. SeaPerch.org distributes construction kits to teachers all across the country, and has played a valuable role in introducing engineering kit projects on the national stage. The Sea Perch ROV construction project has also been adopted by the United States Navy Cadets Corps, and the project has become a staple in many public and private schools throughout New England.

This Sea Perch ROV has been outfitted with an underwater camera for this deployment.

Sea Perch ROV Teacher Trainings

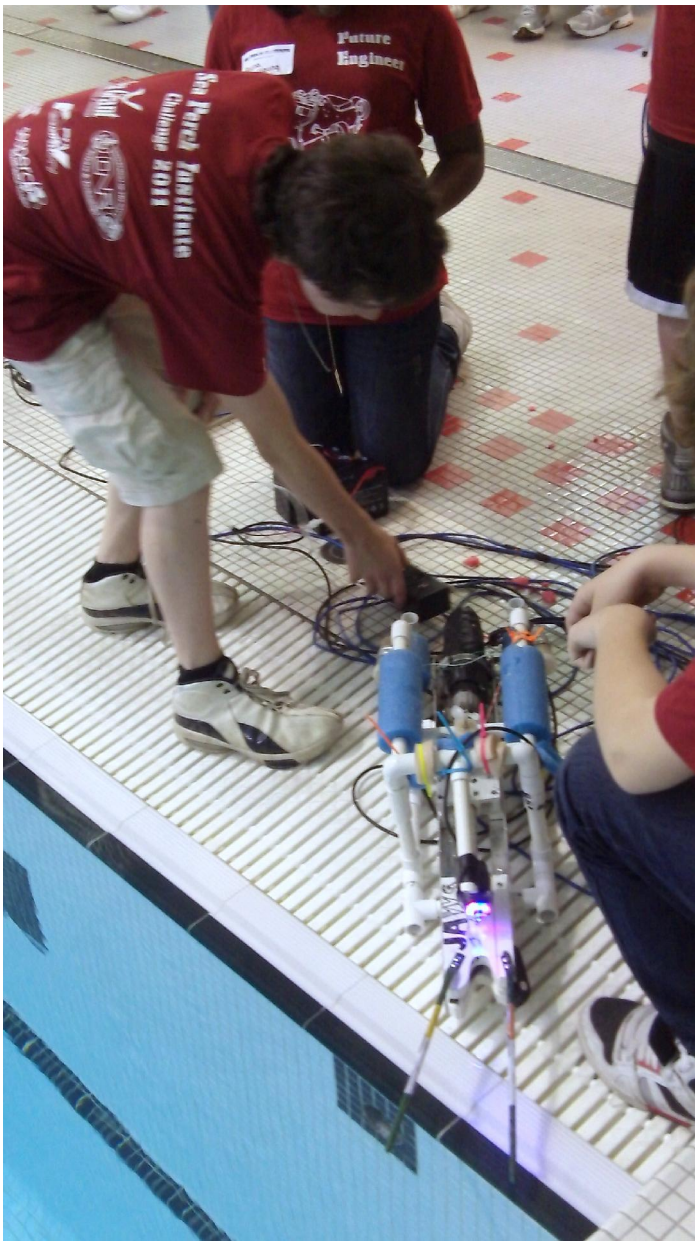
Sea Perch ROV teacher trainings give educators a chance to build the Sea Perch ROV through a multi-day workshop. In this workshop, MIT Sea Grant staff introduce the parts of a Sea Perch ROV: common supplies easily found at a hardware store, like PVC pipe, switches, motors, and a battery. Educators are also introduced to the necessary tools and proper procedures for tool use. Once educators have worked their way through the construction manual and finished the construction of the three units of a Sea Perch ROV (frame, thrusters, and control box) the educators head to an on-site testing facility to run their vehicles.

MIT Sea Grant educators found that the Sea Perch ROV was most effective as an educational STEM project when students participated in a larger-scale project than the simple test deployment shown here.

During the on-site-testing portion of the training, educators pilot the vehicle and learn how to troubleshoot common problems. In this type of training, educators are provided with a brief introduction to extension activities for the Sea Perch ROV, including adding sensing equipment,

such as cameras or water samplers, or augmentations, like grabber arms.

The Sea Perch ROV build process alone has been successfully used to teach basic engineering skills and physics concepts. But it's the extension activities that inspire creativity and meaningful connections between technology, science, and personal experience.



Students from Swampscott Middle School, Massachusetts test the cutting tool on their Sea Perch ROV before deploying it on a mission during the Sea Perch Institute Challenge 2011.

Teaching Engineering Design with Sea Perch Institute

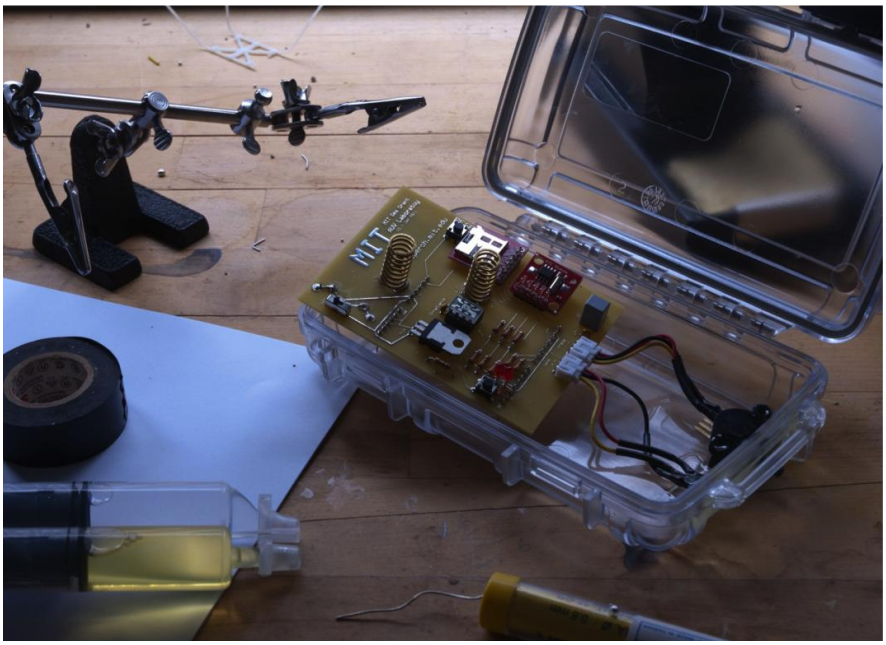
In 2009, MIT Sea Grant began developing the Sea Perch Institute (SPI), a program that encourages students and teachers to move beyond the basic Sea Perch ROV build. SPI is an innovative teacher training course and a project-based, hands-on STEM program for students. Under the mentorship of MIT Sea Grant educators and engineers, teachers who are enrolled in the year-long program work with their students to design, prototype, and implement a solution to a directed ocean engineering challenge. At the end of the school year, teams are invited to the MIT campus for the Sea Perch Institute Challenge final event. Throughout the program, MIT Sea Grant provides technological expertise and shepherds the teachers and students through the complex engineering design process.

The program was piloted during the 2009-2010 school year with four New England schools. During this program year, the students were presented with a complex scenario: a search and salvage mission that required teams to use their Sea Perch ROVs for mapping, water quality analysis, and recovery of cargo and debris. As with all SPI challenge events, the participating schools had to work progressively, cooperatively, and against the clock to complete the mission. During the 2010-2011 school year, teams were charged with developing an emergency response strategy to a surface contaminant and resolution of an underwater "oil pipe" blow out. To tackle this challenge, student teams developed tools for underwater visualization and temperature sensing, and cutting away and removing debris.

Anecdotal evidence suggests that the open, project-based challenge creates a forum in which students are able to grapple with otherwise abstract concepts, such as iterative design.

Open Hardware Instrumentation for Sea Perch ROV

In response to the need for a companion, instrumentation project for use on the Sea Perch ROV, MIT Sea Grant is currently developing and distributing the third version of a microcontroller-based sensor suite. The instrumentation build project is ideal for teaching the core electronic and software principles required to understand the mechanics of oceanographic sensing. In its basic form, the sensor will monitor water temperature, depth, light, and time. The project can also be expanded to allow for additional sensors or instrumentation designed by students.



A completed Sea Perch Sensor Suite version 3, beta.

Like the Sea Perch ROV, the sensor is a build-it-yourself project that provides students with a tangible application for basic physics, and an introduction to computer science. The project uses simple electrical components supported by an intuitive computer language and an abundance of online documentation. The sensor was designed to be sturdy and easy to build, in order to encourage participation from students who have little or no experience with electrical engineering concepts.

Future Work

MIT Sea Grant's Sea Perch ROV has been well received by the STEM education community and students. The open, publicly available build instructions that accompany all of our STEM education technologies are central to the success of the program. In addition, the sandbox-like nature of the Sea Perch ROV build, where students are asked to redesign, augment, and share their work, encourages student creativity, and creates an opportunity for student engagement. Long-term, MIT Sea Grant hopes to develop and distribute materials that would allow the Sea Perch Institute to be

adopted and modified by other industrial partners and universities.

Other future goals include release of project schematics, and a more in depth assessment of the value of the program, particularly with regard to the impact of an open program model. --

Rachel VanCott is the Educator and Ocean Literacy Communicator for the MIT Sea Grant College Program. She is the project manager of MIT Sea Grant's marine science education programs and materials.

Brandy M. M. Wilbur is the STEM Coordinator for Swampscott High School, and an Education Consultant for the MIT Sea Grant College Program. Her focus is transferring technology and education programs to a national audience.

Chryssostomos Chryssostomidis, Director of the MIT Sea Grant College Program, is Doherty Professor of Ocean Science and Engineering, and Professor of Mechanical and Ocean Engineering. His research interests include design methodology for ships, vortex-induced response of flexible cylinders, underwater vehicle design, and design issues in advanced shipbuilding, including the all electric ship and T-Craft.

Kathryn Shroyer is the Engineering Educator for the MIT Sea Grant College Program. She is the director of Sea Perch Institute, and the project manager for Sea Perch ROV technology development, as well as MIT Sea Grant's other ocean engineering programs and materials.

Michael Soroka is a Research Engineer at the MIT Sea Grant AUV lab. He is responsible for AUV design, fabrication and deployment and serves as a technical advisor for the Sea Perch Institute and other MIT Sea Grant education technology projects.

Two students from Swampscott middle school make last minute adjustments to their modified Sea Perch ROV during the Sea Perch Institute Challenge 2011.

Resources

Sea Perch: http://seaperch.mit.edu
MIT Sea Grant : http://seagrant.mit.edu
Office of Naval Research: http://www.onr.navy.mil/
Build Your Own Under Water Robot and Other Wet Projects by Harry Bohm and Vickie Jensen. ISBN 0-9681610-0-6. http://www.westcoastwords.com/books_USD/details.php?book_ID=101

Open Hardware Needs Your Help

Papers

We need stories, on a really short deadline, for the next issue. Please see *Call for Papers* in this issue.

Catalog Wiki Entries

We are constructing a global catalog of Open Hardware projects at http://wiki.openhardware.org/Catalog We need you to help us by adding entries to the catalog. Please go to http://wiki.openhardware.org/Project:How_to_Create_Content_for_the_Open_Hardware_Catalog At that URL, you will find instructions on how to add content to the catalog. Please check if all of the Open Hardware projects you know of are in the catalog, and add content for any that aren't.

Vendor List Wiki Entries

We are also creating a list of vendors that sell parts, Open Hardware kits, any supplies you've used for Open Hardware projects. It's at http://wiki.openhardware.org/Category:Vendor We need you to add all of the places you buy parts to that list. Add an entry through http://wiki.openhardware.org/Add:Vendor_Entry

All of the catalog content you add will be licensed under Creative Commons Attribution-ShareAlike 3.0 and copyrighted by the Open Hardware organization.

An Open Hardware Platform for USB Firmware Updates and General USB Development

By Andrew Stone <stone@toastedcircuits.com>

A key component of open hardware is the ability to update the software within the device. Without that feature, the software in devices cannot be modified. Additionally it is important to understand the true costs of this kind of feature if we are to undermine the tendency of device manufacturers to sell "development kit" functionality for hundreds or thousands of dollars. And future open hardware licenses may require firmware updatability – so there is a need to understand the effort involved so that reasonable exemptions and limits on the cost of external device programmers can be set.

This project provides the hardware design and software library to implement firmware upgrades and general USB access, as a serial port or a human interface device (HID). A major design goal was to implement the cheapest, smallest solution possible so that it can be included in PCBs with little impact. The solution, including the USB port, currently fits on a thumbnail-sized section of a PCB, and has component costs of about $4, in small quantity. It also contains logic to arbitrate the power to the board between its USB port and its DC jack (the jack is preferred), and a automatically-resettable fuse to protect against power overdraw. It is currently in use in the *Lightuino* 5 LED-driver circuit board, and is licensed under the GPL. Also provided is a "stand-alone" design that can connect to the Arduino ICSP port, SPI, i2c, or GPIOs. It can therefore be used to "USB-enable" other simple hardware designs. To demonstrate this functionality, the design also includes some extra LEDs and an IR-receiver for experimentation with USB-HID (human interface device) devices.

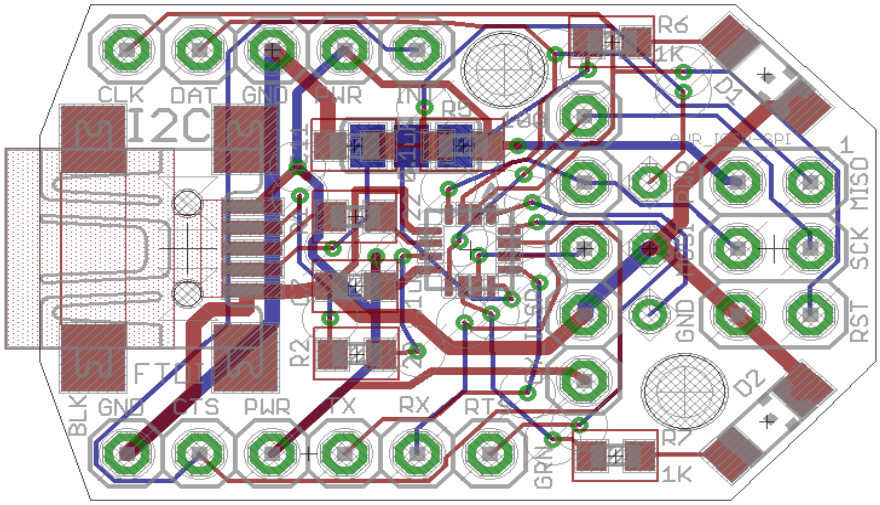


Illustration 1: Board layout for the standalone programmer

For example, as an April fool's joke, the board can be hidden on the USB port in back of a desktop computer and "inject" fake keystrokes like "all your base belong to us" or "I'm sorry, Dave. I'm afraid I can't do that." whenever it receives a command from an IR remote.

Basic Features

Hardware:

- USB mini connector – device only (USB on-the-go is not supported).
- Cypress CY7C64316 full-speed USB chipset with 32k flash.
- USB activity LED.
- IR-receiver.
- PTC fuse to protect upstream USB against power overdraw.

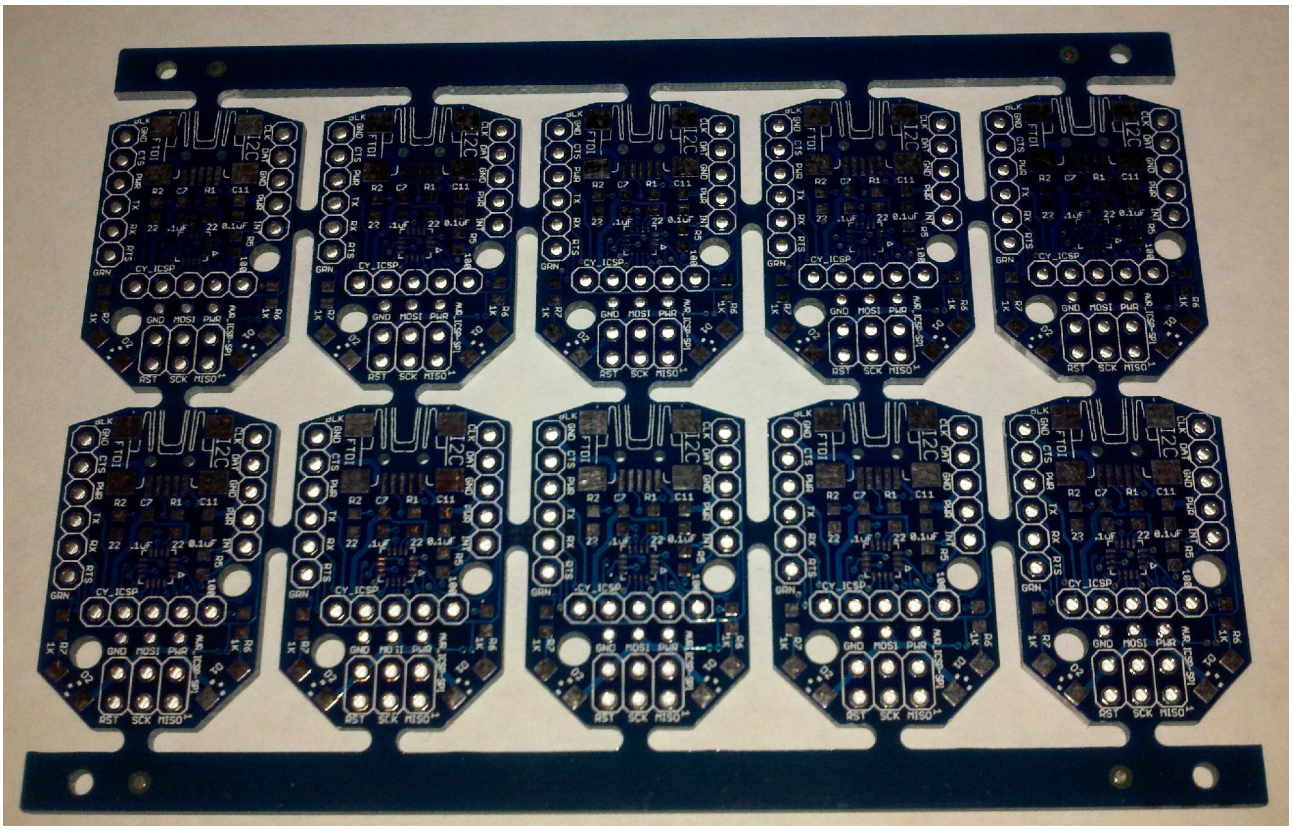


Illustration 2: 10 copies of the standalone programmer professionally fabricated

- Implemented on a 2 sided fine-pitch (professionally made) PCB. But all components are located on a single side for ease of assembly.

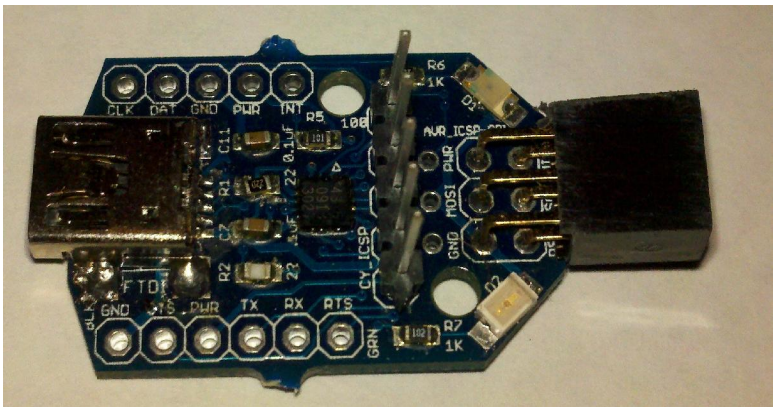


Illustration 3: Standalone programmer fully assembled

Software:

- Implements a "standard" CDC ACM USB serial port which is automatically detected under Linux and can use the standard windows serial.sys driver.
- Converts serial communications to the SPI protocol for forwarding to the embedded CPU (hardware also supports i2c). An Arduino-style library is also provided for the embedded CPU side.
- Implements the AVR bootloader protocol (stk500, the same protocol as the Arduino bootloader) on the USB chip and converts it to ICSP (in-circuit-serial-programming) commands. This allows the full flash of the embedded CPU to programmed state, and minimizes the likelihood of "bricking" your CPU.
- Seamlessly works with the Arduino development environment and the *avrdude* device programming software.

Location

This project is hosted on github at: https://github.com/gandrewstone/toastedCypressUsb. At this point, I can also provide PCBs and component "kits" for the standalone programmer for people interested in prototyping their solution. Please contact me at stone@toastedcircuits.com or through github.

A Note on Open Source Hardware

This solution represents a compromise between open source hardware ideals and the reality that device manufacturers provide today. In this particular project, cost was considered more important then how "open" the device manufacturer's tools are. However, Cypress is not unusual in their use of closed-source tool-sets; this is unfortunately the norm today, meaning that most if not all open source hardware is built on at least some closed-source components or tool-sets.

Software

The solution relies on free Cypress libraries which are license-locked to Cypress chips only. These libraries use direct register access to the Cypress CPU and are entirely written in assembly language, making them essentially unportable to other platforms anyway.

However, the software I provide has been carefully built (almost entirely in C) on top of an "underface" API. So it remains portable to other platforms if the basic SPI and USB underface libraries are provided for the new platform.

Tool-chain

The USB cpu must be compiled under the Cypress PsoC Designer software which is closed-source and can only be hosted on Windows.

Hardware

Programming the Cypress CPU requires a PsoC MiniProg (closed hardware) device, which is available from Cypress for less then $30.

A Note on Surface Mount Parts

This solution uses fine pitch surface mount parts to minimize the cost and PCB area. These parts are sometimes intimidating to hobbyists and other DIY engineers. However, it is very possible to solder these parts using inexpensive home-brew equipment – in fact it is arguably easier and faster to solder a bunch of QFN-32 parts then to individually solder 32 thru-hole pins. And components are increasingly only available in these form factors. Therefore it is extremely important for DIYers to learn these techniques. Many resources and tutorials exist online including my own workshop described here[3].

3 http://effluviaofascatteredmind.blogspot.com/2011_09_01_archive.html

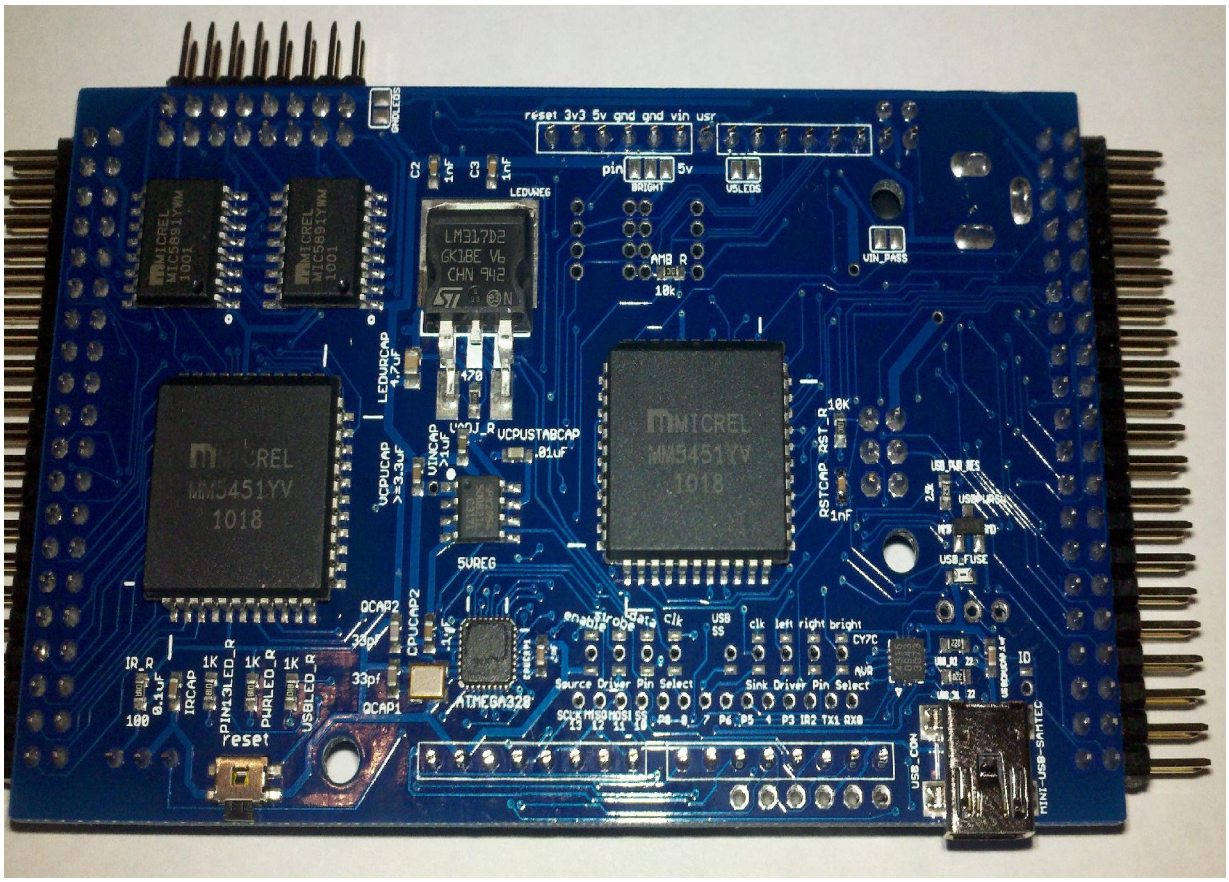


Illustration 4: This board was home-assembled. Note it is almost entirely SMT and contains chip-scale QFN-32 and QFN-24 packages. The USB portion (the subject of this article) is in the lower right, just above the USB connector. The "Arduino-compatible" AVR CPU portion is about 1-cm squared and located in the bottom center.

Comparison with Other Choices

The USB solution on the Arduino is another excellent possibility for firmware upgrade. Its major advantage is that it is based upon the AVR chipset which is more “open development” friendly. For example, toolchains are based on gcc and run on Linux and Linux-based device programmers exist. However, I think that it is likely a bit more expensive; requiring more components and PCB space. And most importantly, on the software side it only implements USB-to-serial connectivity; a bootloader on the main AVR cpu is still required to program the device. On the other hand, this is likely only a software choice dictated by the desire to have strict compatibility with older Arduino boards.

And the “teensy” (http://www.pjrc.com/teensy/) is also a good choice, although it is likely even more expensive as its CPU is a full-featured AVR.

Known Issues

It is possible to configure the AVR 328p, via fuse bits, with such a slow clock that the Cypress CPU can no longer communicate with it over ICSP or SPI. To avoid accidentally doing this, the software refuses to allow those fuse bits to be set.

If running on USB power and a DC jack is inserted, power is briefly lost.

A Return to Open Devices

By Andrew Stone <stone@toastedcircuits.com>

As recently as 30 years ago “stuff” – both the tools required to make a living and leisure and entertainment toys – was a lot simpler. Let's consider, for example, a shovel as a typical tool. It has a handle, a blade and some mechanism to attach the two. In other words, the device's function and construction was transparent to the owner. This transparency implicitly transferred certain rights and responsibilities that were not even recognized as such at the time. These essential rights include:

The right to repair – if the shovel's wooden handle breaks, you do not need to throw away the relatively expensive (in carbon footprint if not in money today) blade. This repair includes using homemade or competitors parts.

The right to modify – you can lengthen or shorten the handle to match your size.

The right to re-purpose – if in the jungle, you can sharpen the blade and make a machete.

The responsibility to make safe repairs and modifications – it's pretty hard to hurt yourself or others by modifying a shovel; other things such as automobiles and lawnmowers are much easier. But once the end user has sharpened that shovel into a machete, the original manufacturer cannot be held responsible for what it cuts. After modification, the owner essentially becomes the “final manufacturer” of the device. To shift this responsibility to an “upstream part supplier” – i.e. the original device manufacturer – the owner must prove that the flaw both existed AND could be triggered in the originally delivered device. In other words, he must prove that the supplied “part” was defective *for the purpose which it was intended to be used*. This releases manufacturers from liability arising from an owners badly conceived modification, and is necessary so that manufacturers are willing to enable device modification.

Prior to the integrated circuit, even simple electrical and mechanical equipment contained enough transparency so that a semi-knowledgeable end user could repair or replace broken parts, and re-purpose. Bicycles[4] and of course automobiles have dedicated hobbyist and collector users. Tools like belt sanders and planers are essentially just an electric motor, a switch, some ball bearings and belts.

However, the situation is rapidly changing. Every device from your blender to your automobile contains embedded processors that are inherent to the correct function of the device. Social interaction occurs on complex personal information devices instead of the traditional paper media. These processors add significant benefits but also come with significant drawbacks. The first drawback is simple -- “bugs” in the code running the systems. The cost to recall and reprogram consumer devices is significant, and development cost and schedule pressures encourage the shipment of devices with problems that may hamper but obviously not completely cripple the device. For example, when you increase the heat setting on my clothes dryer, the estimated drying time increases. But extra heat ought to reduce drying time. Over the 10 years I have used the dryer, this bug has probably consumed significant amount of extra electricity, and this dryer is a very

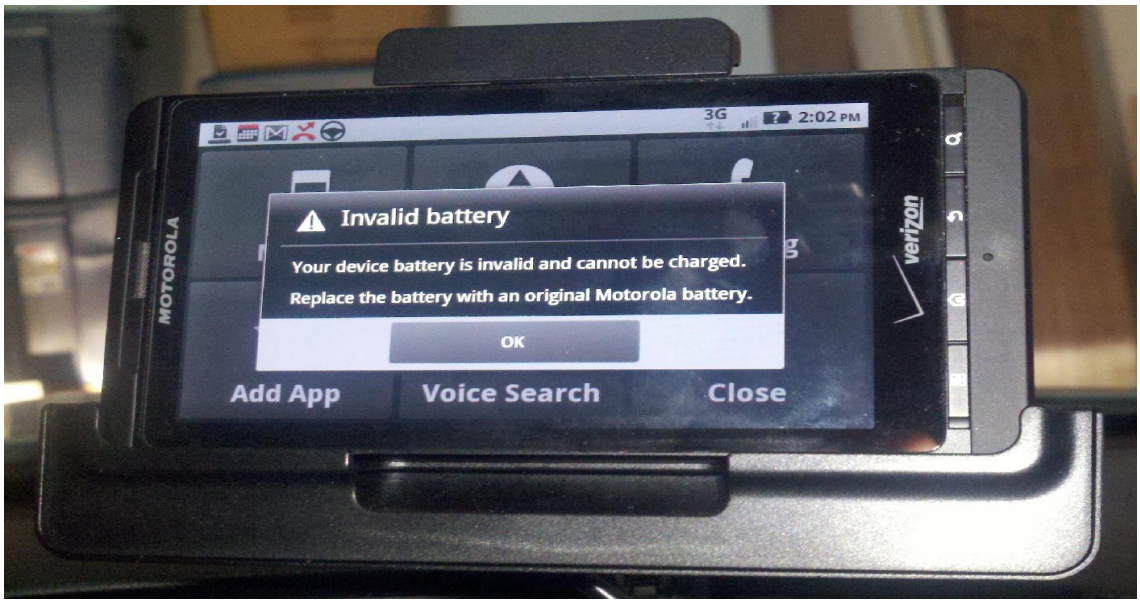
4 http://www.bikeforums.net/forumdisplay.php/225-Alt-Bike-Culture

popular name-brand model.

In fact once you start looking, issues are easy to spot. I have found bugs in microwave ovens (not turning off), stereo receivers, blu-ray players, cell phones (their bugs are *rampant*), automobiles (including a total dashboard reboot at highway speeds!), airline terminal flight display systems, and more.

These problems are unrepairable today by anyone other than the original manufacturer. However, if the firmware were to be made available for modification, the problems could be fixed. While repairing firmware may be significantly more complex then adding a handle to a shovel, this is mitigated by the fact a single user could repair the problem for everybody, by uploading his fix to a common location. In fact, this could result in a broad community of modifications and repairs benefiting all users, and even the original manufacturer who could incorporate community fixes and features into subsequent manufacturing runs. In fact, this very sequence of events happened with the Cisco/Linksys WRT54G home router and several projects, OpenWRT, dd-wrt, and Tomato[5]

Here is a real-life example where this Motorola phone is attempting to "lock me in" to Moto batteries.



I believe that this violates my implicit right to repair objects I own. But the real problem is that this photo was taken with a Motorola battery installed – in fact, *this is the original battery that shipped with my phone*. There is a trend in devices to protect the interests of the manufacturer rather then the interests of the owner/end user. Unfortunately, as we saw with the market failure of Windows Vista and see here, these attempts often make the device inconvenient or unusable by the owner/end user.

All devices have basic protections as general safety measures against casual mistakes. For example,

5 http://en.wikipedia.org/wiki/OpenWrt, http://www.dd-wrt.com, and others

electrical wires are insulated and electric saws often contain momentary (spring loaded) switches that will stop the saw if the hand is removed. However, these protections are merely good design; they would not stop an intentional effort to defeat the safety measure, as might be required to re-purpose the device.

But the modern embedded processor has created a new possibility: it lets a device protect itself against modifications and re-purposing. This facility has generally been used to attempt to curtail the illegal activities of a minority of users (such as with media pirating), or to create vendor lock-in to force additional sales on the razor-and-blades model (printer inks[6] and phone batteries). Unfortunately, these techniques generally inconvenience a majority of end users, discourage healthy competition, and disallow modification and reverse-engineering – by doing so, they deter both innovation and education.

Is it the responsibility of a device to enforce legal activity, or that of law enforcement?

Now that the facility exists to remove essential and previously implicit rights and responsibilities associated with ownership, it is necessary that we move to protect these rights and make these responsibilities clear. We cannot legally force device manufacturers to protect these rights. However, as engineers we can provide alternatives that do so. And as knowledgeable consumers we can make purchasing decisions that are wiser in the long run.

This is the crux of the Open Hardware movement and can be loosely defined as standing on 3 pillars:

1. Open Source Hardware (OSHW): All physical designs and schematics must be freely available to the public under a license that allows modification and resale of new devices.
2. Open Source Software (OSS/FOSS): All software and firmware must be end-user modifiable and compilable, using free (and preferably Open Source) toolchains.
3. Preservation of ownership rights and responsibilities: Including the right to repair, modify, and re-purpose. The legal and social responsibility to do to safely.

To comply with this final requirement, devices must:

1. Allow all software/firmware to be uploaded by the end-user, essentially for free.
2. Be able to be disassembled (and reassembled) by the end-user, into preferably COTS component parts.

Where are we going?

The previous pages have mostly focused on the negative by highlighting issues with closed devices and arguing that what we do not like about them is driving us towards the Open Device. But there

6 http://news.cnet.com/2100-1023-979791.html

are also significant positive forces moving us there.

For example, a smart phone – which is essentially a tiny and very complex computer with a lot of complex RF circuitry -- requires significant development time by a group of highly specialized engineers. But given an open smart phone, anyone could add functionality to it aligned with their speciality – people will build geiger counters, health monitors, chemical sensors, biological assay devices, remote monitoring stations. Previously, each device needed its own screen, buttons and embedded CPU, which are entire specialities in themselves.

So tomorrow's open smart phone can act as the user interface and brain for a huge ecology of diverse handheld "snap-on" hardware. Although today's Android phones are not Open Hardware, they are mostly Open Source software and programmable platforms, and can be interfaced through bluetooth and USB. With these phones, the first steps in the direction I propose have already occurred with Google's "Open Accessory Standard". [7] Physically phones had bluetooth and of course USB interfaces before this standard. The true announcement is not technical or physical, it is social. It is the transformation of the perception of the phone away from a monolithic entity towards a device that exists within a interconnected set of personal devices.

Whenever a period of change arrives, it is interesting and frankly amusing retrospectively to project how the world will have changed once the transformation runs its course; say a decade from now. These guesses will help explain how Open Hardware can be a positive force in our society.

Component Manufacturers

Component manufacturers (chip makers) will need to change the way they do business; they will no longer sell "development kits" at ten to 100 times fabrication cost; they will realize that actually having individuals using their development kits pays itself back in Open Hardware designs and in mind-share (which ultimately creates additional product sales). This is already happening through competition with the Arduino platform[8].

Secondly their entire method of software production will change. Today they attempt to produce software "appnotes" featuring for example, implementation of a protocol. This code is often FOSS but license-locked to their hardware platform, and in my experience is often not product-ready – it is merely a sales tool, a proof-of-concept unfit for deployment. This creates a market inefficiency where every chip manufacturer must produce the same protocol and every device maker must fix the same bugs (and learn different APIs for the same protocol on each chip). With the existence of an open device containing the same protocol, they will realize that the cheapest, fastest, and most valuable solution is to simply port that implementation's "underface" (how it talks to the hardware) to their chip.

Third, component manufacturers will financially support OSHW efforts, recognizing that they are a much better sales tool then internal appnotes or development boards.

7 http://www.engadget.com/2011/05/10/google-announces-android-open-accessory-standard-arduino-based/
8 http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_(MSP-EXP430G2)

Device Manufacturers

The existence of reciprocally-licensed OSHW will cause traditional device manufacturers to polarize into "open" and "closed" varieties. Just like today's "i" stuff, the "closed" varieties may provide a more uniform and carefully constructed experience but will lack the functional breadth of their "open" cousins.

A lot more goes into a product than just the engineering plans, which will allow "open" manufacturers to build a business around an open product even though theoretically anyone could produce the same device. Community-driven R&D will massively reduce their expenditures, as will high-volume production runs to satisfy a large community. These higher volumes and established relationships with retail stores will ensure that the company can continue to undersell new competition but at the same time the fact that the device is open will stop monopoly-style pricing. Also, there may be fewer market failures, as the OSHW community will act as a test market, will more accurately build what people want, and will modify existing devices to BE what is wanted.

Consumers

Consumers will have greater choice in products, initially between "closed" or "open" devices. And within the "open" device category, there will be a "wild-west" of variants, software customizations, and hardware add-ons. Shieldlist.org for example counts over 250 Arduino add-on boards[9]! And most importantly, consumers will have the ability to contribute to and modify the devices they own; either by actually doing the work, or by voting with their dollars in marketplaces like kickstarter[10].

In Closing

We are on the brink of a transformation in how physical devices are created and who creates them. The ability to create modern electronic devices is spreading from the hands of large-budget R&D groups of specialized engineers to communities made up simply of the people who want to be involved; engineers, artists, and other creative types[11]. This transformation will ultimately produce the devices that people want and need, not what focus groups and corporations think is needed. And most importantly this change will transform us into a society of citizen-makers – a people with greater understanding of the things we make and that make us who we are.

9 http://shieldlist.org/

10 http://www.kickstarter.com/

11 For the effects on software, see: http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar

Call for Papers

Deadlines

November 21, 2011 for the December 1 issue.

December 15, 2011 for the January 1 issue.

January 1, 2012 for the February 1, 2012 issue.

Email finished papers or correspondence to bruce@perens.com

Papers must be on the topic of Open Hardware. The licensing of the Open Hardware must be compliant with the Open Hardware Definition 1.1 (as it existed on August 2011), at http://freedomdefined.org/OSHW_draft . Design files must be available, please provide the links in your article.

We'd be delighted if you'd use LibreOffice and its OpenDocument file formats to write your article, as that's what we're using. You can also use LibreOffice to convert your article from *Microsoft Word,* etc. If it's too much trouble to use LibreOffice, please use whatever you wish and we'll convert the file. We strongly prefer an editable file format to PDFs.

Articles will remain your property or that of your institution, as you decide among yourselves, and you will retain all rights of a copyright holder, including the right to reprint as you like with no fee to us.

If you don't understand how to license your article, we'll do it for you. All articles must be licensed as specified below.

- The Creative Commons Attribution 3.0 United States (CC BY 3.0) at http://creativecommons.org/licenses/by/3.0/us/

How to Copy This Journal

Simple Copying Permission

You are welcome to print, copy and redistribute exact copies of this journal, as long as you don't charge a fee. Please use the copy at http://OpenHardware.org/journal/ as this will always be a correct version.

Simple Translation and Reformatting Permission

You may translate this journal, or reformat it for presentation in another file format or on a particular kind of display device. You may not charge for copies. You must include a statement that this is a translated or reformatted version, identifying yourself and providing your contact information. You may print, copy, and redistribute the modified version under the same terms as the original. The modified version must be faithful to the content of the original – don't remove content or add your own other than your attribution as translator. Your modifications become the property of the copyright holders of the original version. Please email translations to the editor at bruce@perens.com, we'll re-publish them on our site.

More Complicated

Most needs will be satisfied by the above paragraphs, without involving a lawyer. You can do more if you read and understand the license statements for the trademark and content, below. You're encouraged to consult your legal counsel. You're also welcome to contact us, we will grant special permissions when appropriate, explain our policies and licenses, etc.

The logo of an integrated circuit chip and a padlock with unlocked hasp is a controlled-use trademark of the Open Hardware organization. You may reproduce it in the exact context in which we used it in this journal. Please do not otherwise reproduce the logo except under authorization of the Open Hardware organization. A contract for use of the logo is under development.

The compilation which is this issue of the Open Hardware Journal is under this license:

- The Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States (CC BY-NC-SA 3.0) at http://creativecommons.org/licenses/by-nc-sa/3.0/us/

We used this license because of the problem of unscrupulous individuals who make our content available for a fee, for example on e-reader download sites, and claim it as their own. We intend for everyone to be able to read our journal at no charge. We left the individual articles under a more liberal license, so that our community doesn't suffer from our attempt to control the unscrupulous folks.

The authors have chosen to license their works under:

- The Creative Commons Attribution 3.0 United States (CC BY 3.0) at http://creativecommons.org/licenses/by/3.0/us/

Optionally, you may extract an individual article from the journal, removing references to the journal, and treat it under the more liberal terms that the author has chosen.

Credits

Our logo is by Laura Rodríguez, know also as "LiR", of Papermint Designs.
Cover photo: Bruce Perens

Colophon

This issue was edited and produced on a *Debian* GNU/Linux system. All of the tools used were Open Source / Free Software. Most of them came directly from the Debian distribution. The paper authors also appear to have used Open Source / Free Software tools.

The main editing tool was LibreOffice.